Exercise 3 – Tapio Koskinen 1800509

1. Explain following terms:
   a) Abstraction

   Abstraction in object oriented programming is showing essential information or attributes and hiding the non relevant. For example an object called user might have a lot of attributes, so with abstraction the relevant can be filtered.

   b) Accessor and mutator methods

   Internal data is often kept private, so the way a program can access private data is to create a class method called accessor, usually named with 'get' something. Mutator is a method that modifies data, like the value of a variable and is usually named 'set' something.

   c) Public and private methods

   In OOP Public method can be used both inside **and** outside of the class. Everything that is **not** marked as private within a class can be called and used. A class that is marked private cannot be called outside of the class. To pass private methods or attributes, they must be handled inside the function.

   d) __str__ method (in python)

   __str__ method, which is defined at the end of a class, returns a string when called. The interesting part is that it can take object values within a string and print the value for example "a {self.__color} {self.__animal}", where color would be blue and animal bird, would print out "a blue bird" instead of the brackets etc.

2. Modify the coin class to have currency attribute and currency generator

   Fist create the currency

```
def __init__(self):
    self.sideup = 'Heads'
    self.currency = 'Euro'
```

Next I created the methods to generate and get the current value

```
    # Create a dictionary, set currency to a random value generated from
    # the dictionary

    def gen_currency(self):
        currency_list = {1:"Euro",2:"Pound",3:"Dollar",4:"Ruble",5:"Yen"}
        self.currency = currency_list[random.randint(1,5)]

    #Return the currenct value of currency

    def get_currency(self):
        return self.currency


# The main function

def main():

    # Create an object from the Coin class.
    my_coin = Coin()

    # Just testin here
    print(my_coin.get_currency())
    my_coin.gen_currency()
    print(my_coin.get_currency())
    my_coin.gen_currency()
    print(my_coin.get_currency())
```

On console(with the coin stuff)

```
= RESTART: C:\
coin_demo.py
Euro
Pound
Pound
This side is up: Heads
I am tossing the coin...
This side is up: Heads
>>>
= RESTART: C:\
coin_demo.py
Euro
Yen
Euro
This side is up: Heads
I am tossing the coin...
This side is up: Tails
>>>
= RESTART: C:\
coin_demo.py
Euro
Yen
Dollar
This side is up: Heads
I am tossing the coin...
The coin dropped into a rabbit hole
>>>
```

3) Choose currency method

```
def __init__(self):
    self.sideup = 'Heads'

    # initial value is Euro
    self.currency = 'Euro'

    # Changed the code so that the currency_list is initialized here
    # so other methods can use it as the please

    self.currency_list = {1:"Euro",2:"Pound",3:"Dollar",4:"Ruble",5:"Yen"}
```

The method

```
# Ask the user for a value between 1-5 and set the
# value of currency to the key value of currency_list

def change_currency(self):
    currency_choise = int(input((f"Choose your currency: \n{self.currency_list}\n")))
    self.currency = self.currency_list[currency_choise]
```

```python
def main():

    # Create an object from the Coin class.
    my_coin = Coin()

    # Just testin here
    print(my_coin.get_currency())
    my_coin.gen_currency()
    print(my_coin.get_currency())
    my_coin.gen_currency()
    print(my_coin.get_currency())

    #Ask user to choose currency and print it
    my_coin.change_currency()
    print("Currency: ",my_coin.get_currency())
```

From the console

```
= RESTART: C:\C_
coin_demo.py
Euro
Pound
Pound
Choose your currency: {1: 'Euro', 2: 'Pound', 3: 'Dollar',
2
Currency:  Pound
This side is up: Heads
I am tossing the coin...
This side is up: Tails
>>>
= RESTART: C:
coin_demo.py
Euro
Dollar
Ruble
Choose your currency:
{1: 'Euro', 2: 'Pound', 3: 'Dollar', 4: 'Ruble', 5: 'Yen'}
4
Currency:  Ruble
This side is up: Heads
I am tossing the coin...
This side is up: Tails
>>>
```

4) Change to private with the dunderscore(double underscore)

```python
class Coin:

    # The __init__ method initializes the sideup data attribute with heads
    # edit: now sideup up is private
    def __init__(self):
        self.__sideup = 'Heads'
```

Trying to get the attribute straight

```
# Display the side of the coin that is facing up.
print('This side is up:', my_coin.__sideup)
```

Gives an error

```
Traceback (most recent call last):
  File "C:\ ____                _   ...  ....\ObjectOrientedProgramming\Lecture2\coi
n_demo.py", line 111, in <module>
    main()
  File "C:\'    ..              _          \ObjectOrientedProgramming\Lecture2\coi
n_demo.py", line 90, in main
    print('This side is up:', my_coin.__sideup)
AttributeError: 'Coin' object has no attribute '__sideup'
>>>
```

5) Class Dice

Pseudocode

```
class dice

    values in start
        side up
        color
        colorlist dictionary

    get side up method
        return side up value

    roll dice method
        dice number = random value between 1-6
        color = compare dice number with dictionary

    get color method
        color = compare dice number with dictionary

    cheat method
        choose dice number
        add color
```

Class Dice, with the extra method cheat

```
# New class Dice

class Dice:

    # Define the data on initalization
    def __init__(self):
        self.sideup = "None"
        self.color = "None"
        self.color_list = {1:"Red",2:"Green",3:"Blue",4:"Yellow",5:"Orange",6:"Purple"}

    # Similar to coin, get the side up
    def get_sideup(self):
        return self.sideup

    # Get the color facing up
    def get_color(self):
        return self.color

    # Similar to coin, but rolling a number and a color
    def roll(self):
        self.sideup = random.randint(1,6)
        self.color = self.color_list[self.sideup]

    # cheat the outcome
    def cheat(self):
        cheat_value = int(input("1-6: "))
        if cheat_value > 6:
            self.sideup = cheat_value%6
            self.color = self.color_list[self.sideup]
        else:
            self.sideup = cheat_value
            self.color = self.color_list[self.sideup]
```

main

```python
# Author: Tapio Koskinen
# File: dice_test.py
# Date: 30.1.2021
# Description: Exercise 3 - part 5,6,7

import Coin_Dice_demo

def main():

    # Set the class into dice
    dice = Coin_Dice_demo.Dice()

    print("With my dice, we can decide which color shirt you can use")

    #Ask user to choose instead of rolling
    choise = input("Wanna choose your destiny?(y/n): ")
    if choise == "y":
        dice.cheat()
    if choise == "n":
        dice.roll()

    # Print the value of the dice
    print(f"The dice drops as: {dice.get_sideup()}")

    #Print the color
    print(f"Your shirt color of the day is: {dice.get_color()}")


main()
```

Console

```
= RESTART: C:\                                .\ObjectOriente
\ExerciseFiles\CodeFiles\dice_test.py
With my dice, we can decide which color shirt you can use
Wanna choose your destiny?(y/n): y
1-6: 2
The dice drops as: 2
Your shirt color of the day is: Green
>>>
```

```
With my dice, we can decide which color shirt you can use
Wanna choose your destiny?(y/n): n
The dice drops as: 2
Your shirt color of the day is: Green
>>>
```

I just now realized the random value is the same as the userinput value, but it does generate random.

6) Two dice and sum

```
    my_dice = Coin_Dice_demo.Dice()
    notmy_dice = Coin_Dice_demo.Dice()

    my_dice.roll()
    notmy_dice.roll()

    print(my_dice.get_sideup() + notmy_dice.get_sideup())
main()
```

Console

```
= RESTART: C:\
\ExerciseFiles\CoderıIes\dice_test.py
8
>>>
```

7) Dice game -- Pseudocode

```
import dice

define main program:

    player1 = dice
    player2 = dice
    player3 = dice

    playerlist = all players

    define the game(playerlist):

        while first round is true
            smallest value = 6

            all players roll

            if player2 value is less than player1
                smallest value = player 2 dice

            else
                smallest value = player 1 dice

            if player 3 value is less than smallest value
                smallest value = player 3 dice
                remove the player with the lowest value from the list
                break from loop

            else if player 3 is equal to smallest value
                continue loop

            else
                remove the player with the lowest value from the list
                break from loop

        while second round is true

            remaining players roll

            if remaining player1 < remaining player2
                remove remaining player 1 from playerlist
                break from loop

            else if remaining player1 > remaining player2
                remove remainig player 2 from playerlist
                break from loop

            else
                continue loop

        print the winner as the only element in playerlist
```

Initial values

```
# Author: Tapio Koskinen
# File: dice_game.py
# Date: 30.1.2021
# Description: Exercise 3 - part 7

# Import the Dice

import Dice_demo

def main():

    #Assign dices with owners to 3 players

    player1 = Dice_demo.Dice("Steve")
    player2 = Dice_demo.Dice("John")
    player3 = Dice_demo.Dice("Alice")

    # First round variable for looping

    first_round = True

    # Make a list to keep track of players

    player_list = [player1,player2,player3]

    print("Time for a dice tournament!")
```

```
# First round
while first_round:

    # Throw for all player
    for i in player_list:
        i.roll()
    print(f"First round rolls:\n{player1.get_owner()}: {player1.get_sideup()}\nJohn: {play

    # Set the initial value of smallest to player 1
    smallest = player1.get_sideup()

    # Set smallest to player2 if its smaller than smallest
    if player2.get_sideup() < smallest:
        smallest = player2.get_sideup()
        #If player 3 threw less, remove player3 from the game and end the round
        if player3.get_sideup() < smallest:
            player_list.remove(player3)
            first_round = False
        #If player 3 dice equals to the smallest, everyone throws again
        elif player3.get_sideup() == smallest:
            print("Roll again!")
            continue
        #Otherwise remove player2 from the game
        else:
            player_list.remove(player2)
            first_round = False

    # Run through checkups if p1 and p2 have same values
    elif player2.get_sideup() > smallest:
        if player3.get_sideup() >= smallest:
            print("Roll again!")
            continue
        else:
            player_list.remove(player1)
            first_round = False
```

```
    # Round 2 begins with remaining players
    round2 = True

    while round2:
        #Just to make it prettier
        p1 = player_list[0]
        p2 = player_list[1]

        # Roll
        p1.roll()
        p2.roll()

        # Announce players rolls using objects

        print(f"Second round rolls:\n{p1.get_owner()}: {p1.get_sideup()}\n{p2.get_owner()}: {p

        #Check for the winner

        if p1.get_sideup() > p2.get_sideup():
            print(f"{p1.get_owner()} wins!")
            round2 = False
        elif p1.get_sideup() == p2.get_sideup():
            continue
        else:
            print(f"{p2.get_owner()} wins!")
            round2 = False

main()
```

Console

```
= RESTART:
Time for a dice tournament!
First round rolls:
Steve: 2
John: 6
Alice: 2
Roll again!
First round rolls:
Steve: 5
John: 1
Alice: 5
Second round rolls:
Steve: 5
Alice: 6
Alice wins!
>>>
```

8) CellPhone – Pseudocode

```
Pseudocode for cellphone

define class cellphone:

    define values on initialization:
        manufacturer
        model
        reailtprice

    define set manufacturer method
        manufacturer = input

    define set model name method
        model = input

    define set retail price method
        retailprice = input

    define get manufact method
        return manufact value

    define get model method
        return model value

    define get retail price method
        return retail price value
```

Class CellPhone

```
"""
Created 30.1.2021
@File: CellPhone.py
@Description: The Cellphone
@author: Tapio Koskinen

"""
# Create a class CellPhone
class CellPhone:

    # Create and set the initial values
    def __init__(self):
        self.__manufact = "Nokia"
        self.__model = "3310"
        self.__retailPrice = 20

    # set a name for manufact
    def set_manufact(self):
        self.__manufact = input("Set manufacturer: ")
    # set a name for model
    def set_model(self):
        self.__model = input("Set model: ")
    # set the retail price
    def set_retailPrice(self):
        self.__retailPrice = float(input("Set retail price: "))
    # Get the manufacturer
    def get_manufact(self):
        return self.__manufact
    # Get the model
    def get_model(self):
        return self.__model
    # Get the retail price
    def get_retailPrice(self):
        return self.__retailPrice
    # Returns all spesifications of the phone
    def __str__(self):
        return f"""\nManufacturer: {self.get_manufact()}
Model: {self.get_model()}
Retail price: {self.get_retailPrice()}"""
```

Main

```
def main():

    my_phone = CellPhone()

    my_phone.set_manufact()

    my_phone.set_model()

    my_phone.set_retailPrice()

    print("Here is the data that you provided:",my_phone)

    my_phone.set_manufact()

    my_phone.set_model()

    my_phone.set_retailPrice()

    print("Here is the data that you provided:",my_phone)

main()
```

Output:

```
= RESTART: C:\ .                .
                          \CellPhone.py
Set manufacturer: Apple
Set model: iPhone7
Set retail price: 500
Here is the data that you provided:
Manufacturer: Apple
Model: iPhone7
Retail price: 500.0
Set manufacturer: Apple
Set model: iPhone 7
Set retail price: 500.0
Here is the data that you provided:
Manufacturer: Apple
Model: iPhone 7
Retail price: 500.0
>>>
```

9)

a) Object: The my_phone is an object

b) Encapsulation: The class Cellphone is an encapsulation

c) Data attributes: Manufacturer, model and retail price

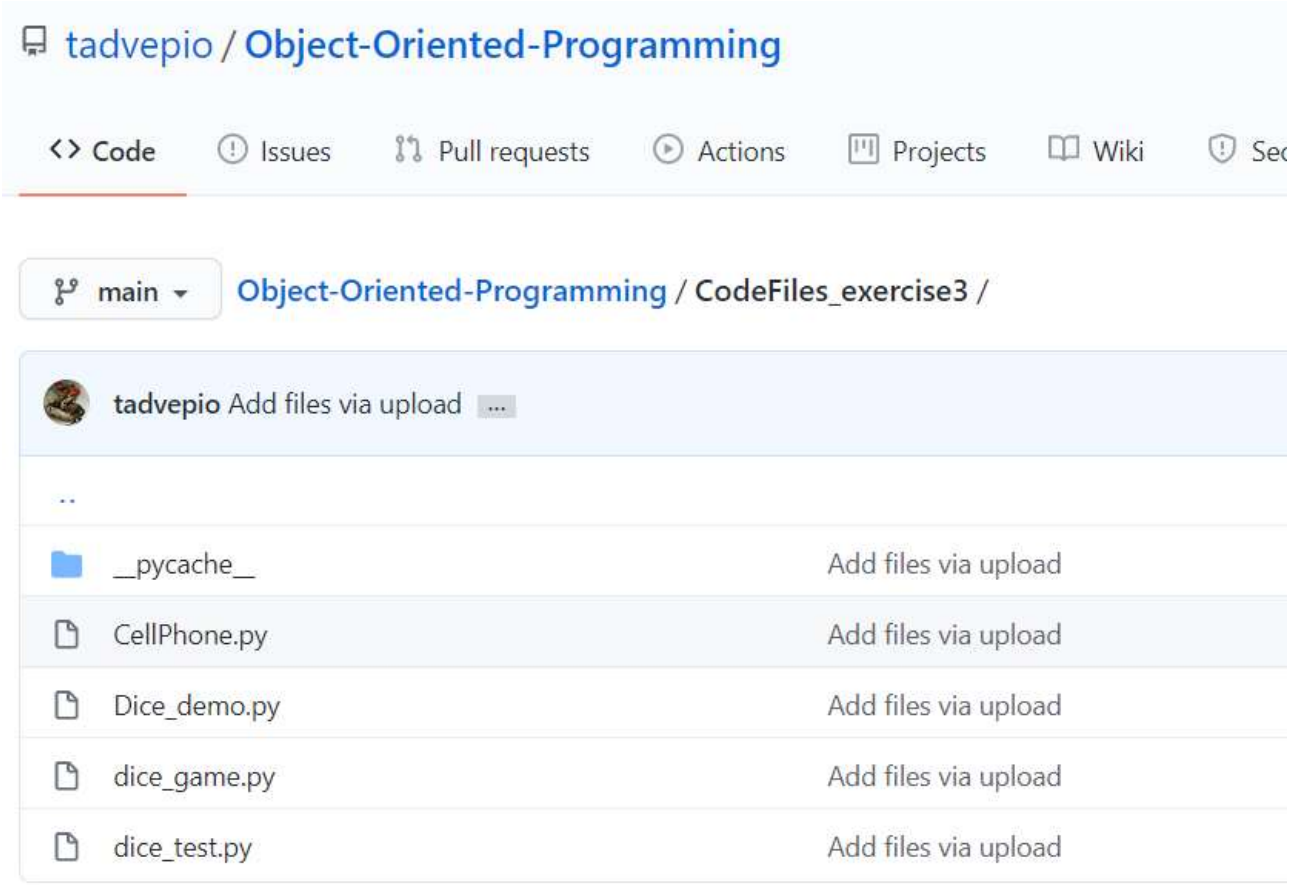
d) Hidded: The ones that are dundered(double underscore)

e) Public methods: All of the methods are public

f) Private methods: None are private, because none are specified private

g) Init method: __Init__ is in the beginning of the code, creating and assigning values to manufact, model, and retailprice.

Screen capture of the Git status:



Self-assessment:

I'm starting to understand the concept of object oriented programming better than in the beginning.

I did have trouble designing the dice game when juggling the values and comparisons and it could have been more elegant, but it works.

I feel more confident than before. This exercise took me the longest so far.